Title: GUI-Based Calculator with Exception Handlin Mechanism

Introduction

This Java program implements a simple graphical calculator using Swing components. The calculator provides basic arithmetic operations (addition, subtraction, multiplication, division, modulus), square root functionality, and standard calculator features like clear and backspace.

Key features of the calculator include:

- Clean, dark-themed graphical user interface
- Support for basic arithmetic operations (+, -, ×, ÷, %)
- Square root calculation (√)
- Decimal point input
- Clear (C) and backspace (⌫) functionality
- Error handling for invalid operations (like division by zero)
- Responsive button layout with color-coded operations

The calculator follows standard arithmetic order of operations (though it doesn't support complex expressions with multiple operations at once) and displays both the calculation expression and result when the equals (=) button is pressed.

Code

```
package com.mycompany.simplecalculator;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

public class SimpleCalculator implements ActionListener {

    JFrame frame;

    JTextField display;

    JPanel buttonPanel;
```

```
    JLabel expressionLabel;
    double num1, num2, res;
    String op;
    SimpleCalculator() {
        frame = new JFrame("Simple Calculator");
        frame.setSize(350, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());
        frame.getContentPane().setBackground(Color.decode("#1C1C2E"));

        JPanel displayPanel = new JPanel(new BorderLayout());
        displayPanel.setBackground(Color.decode("#1C1C2E"));
        JLabel label = new JLabel("Simple Calculator",
SwingConstants.RIGHT);
        label.setFont(new Font("SansSerif", Font.BOLD, 14));
        label.setForeground(Color.WHITE);
        label.setBorder(BorderFactory.createEmptyBorder(10, 10, 0, 10));
        frame.add(label, BorderLayout.NORTH);
          // Expression label to show the input history
        display = new JTextField();
        display.setFont(new Font("Arial", Font.BOLD, 24));
        display.setEditable(false);
        display.setBackground(Color.decode("#1C1C2E"));
        display.setForeground(Color.WHITE);
        display.setHorizontalAlignment(SwingConstants.RIGHT);
```

```
        display.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        frame.add(display, BorderLayout.CENTER);
        buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(5, 4, 10, 10));
        buttonPanel.setBackground(Color.decode("#1C1C2E"));
        buttonPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10));

        String[] buttons = {
            "C", "⌫", "%", "÷",
            "7", "8", "9", "×",
            "4", "5", "6", "−",
            "1", "2", "3", "+",
            "0", ".", "√", "="
        };
        for (String text : buttons) {
            JButton btn = new JButton(text);
            btn.setFont(new Font("Arial", Font.BOLD, 20));
            btn.setFocusPainted(false);
            btn.setForeground(Color.WHITE);
            switch (text) {
                case "C":
                    btn.setBackground(Color.YELLOW);
                    btn.setForeground(Color.BLACK);
                    break;
```

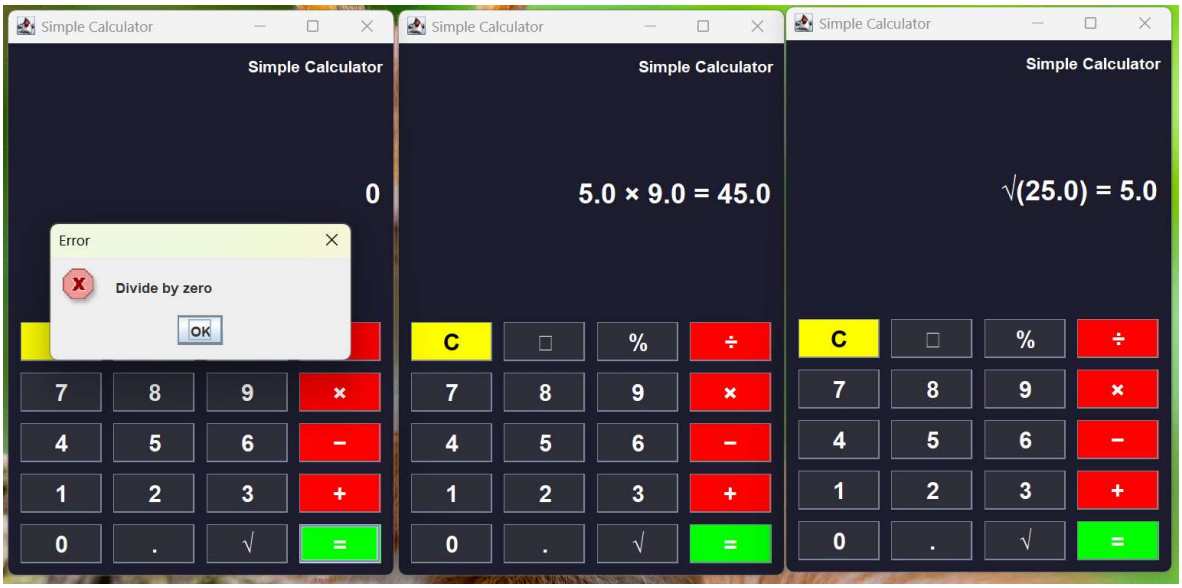
```
            case "=":
                btn.setBackground(Color.GREEN);
                break;
            case "√":
                btn.setBackground(Color.decode("#2F2F3B"));
                break;
            case "⌫":
            case "%":
            case ".":
            case "0": case "1": case "2": case "3":
            case "4": case "5": case "6": case "7": case "8": case "9":
                btn.setBackground(Color.decode("#2F2F3B"));
                break;
            default:
                btn.setBackground(Color.RED);
        }
        btn.addActionListener(this);
        buttonPanel.add(btn);
    }
    frame.add(buttonPanel, BorderLayout.SOUTH);
    frame.setVisible(true);
 }
public void actionPerformed(ActionEvent e) {
 String cmd = ((JButton) e.getSource()).getText();
 if (cmd.matches("[0-9]")) {
```

```
        display.setText(display.getText() + cmd);
    } else if (cmd.equals(".")) {
        if (!display.getText().contains(".")) {
            display.setText(display.getText() + ".");
        }
    } else if (cmd.equals("C")) {
        display.setText("");
    } else if (cmd.equals("⌫")) {
        String cur = display.getText();
        if (!cur.isEmpty())
            display.setText(cur.substring(0, cur.length() - 1));
    } else if (cmd.equals("%") || cmd.equals("÷") || cmd.equals("×") ||
cmd.equals("−") || cmd.equals("+")) {
        if (!display.getText().isEmpty()) {
            num1 = Double.parseDouble(display.getText());
            op = cmd;
            display.setText(""); // Clear for next input
        }
    } else if (cmd.equals("√")) {
        if (!display.getText().isEmpty()) {
            try {
                double val = Double.parseDouble(display.getText());
                if (val < 0) {
                    throw new ArithmeticException("Can't root negative
number");
```

```
            }
            double result = Math.sqrt(val);
            display.setText("√(" + val + ") = " + result);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Invalid input", "Error",
JOptionPane.ERROR_MESSAGE);
        } catch (ArithmeticException ex) {
            JOptionPane.showMessageDialog(frame, ex.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
  } else if (cmd.equals("=")) {
    if (display.getText().isEmpty()) return;
    try {
      num2 = Double.parseDouble(display.getText());
      String expression = "";
      switch (op) {
        case "+":
          res = num1 + num2;
          expression = num1 + " + " + num2;
          break;
        case "−":
          res = num1 - num2;
          expression = num1 + " − " + num2;
          break;
        case "×":
```

```
                res = num1 * num2;
                expression = num1 + " × " + num2;
                break;
            case "÷":
                if (num2 == 0) throw new ArithmeticException("Divide by
zero");
                res = num1 / num2;
                expression = num1 + " ÷ " + num2;
                break;
            case "%":
                if (num2 == 0) throw new ArithmeticException("Mod by zero");
                res = num1 % num2;
                expression = num1 + " % " + num2;
                break;
        }
        display.setText(expression + " = " + res);
      } catch (ArithmeticException ex) {
        JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
      }
   }
}
   public static void main(String[] args) {
      new SimpleCalculator();
   }
}
```

Output



Limitations

1. **Single Operation Limitation**: The calculator can only perform one operation at a time (e.g., 5+3). It cannot handle complex expressions like (5+3)*2.
2. **No Operator Precedence**: Since it only handles one operation at a time, operator precedence rules are not implemented.
3. **No Memory Functions**: The calculator lacks memory storage functions (M+, M-, MR, MC) that are common in standard calculators.
4. **No History Feature**: Previous calculations are not stored or displayed for reference.
5. **No Scientific Functions**: Advanced mathematical functions (trigonometric, logarithmic, etc.) are not included.
6. **Fixed Window Size**: The calculator window is not resizable, which might not be ideal for all display sizes.
7. **No Theme Options**: The dark theme is hardcoded with no option to change colors or appearance.